Automated Modelization of Dynamic Systems

Ivan Perl, Aleksandr Penskoi
ITMO University
Saint-Petersburg, Russia
ivan.perl, aleksandr.penskoi@corp.ifmo.ru

***Abstract*—Nowadays, different kinds of modelling settled down in most areas of human activities. Often, it is not easy to design and create a model of an existing systems, because equations linking its components are not known. This article describes an approach allowing to identify system components interaction based on run-time monitoring and analysis of system behavior. Described method is shown on a simple system that will be turned in to a system dynamics mode by automated equations definition and results will compared to initial analytic model for the same system.**

I. Introduction

We are living in a complex world and with new technological and scientific achievements our world became even more complex, almost, on a daily basis. That is why modelling taking leading positions when it comes to question of systems analysis and design.

There are many different classes of systems and, consequentially, number of ways to design, define, build and compute related models. One of the most frequent kinds of systems we are seeing around are dynamic systems, or systems, behaving or evolving over time. Approach to work with kind of systems is called System Dynamics. Appeared initially in mid 60s and designed for describing economics processes, later, in terms of growth of its methodological maturity, became applicable to work with almost any dynamic system.

System dynamics models consist of two key elements: stocks and flows. Where stocks are representing accumulation of anything accounts receivable, capital, delays, inventories, and so on and flows are representing movement of anything consumption, deliveries, expenses, production, etc.[1]

Today, computer modelling, like many other services, is following a trend of moving to the cloud. There are many benefits of using cloud services, comparing to maintenance own deployments of required systems and services, but in a light of discussed topic, key of them are computation capacity and simplified data access. In most cases, cloud services, especially designed for computation and modelling needs, are hosted on a specialized and well configured equipment, which is fine-tuned for solving particular types of tasks. Because of various kinds of optimization's, such hosted solution will be more efficient even comparing with the same hardware hosted locally, but without specific fine-tuning. That is why cloud services can do more and do this cheaper. Second aspect is data access. When dealing with cloud services there are no issues with bubbling data up to the client or with getting them back or sharing with colleagues and your research fellows.

When it comes to systems modelling these two benefits starting to play even more valuable role together. Having efficient model execution engine, taken in parallel with monitoring data of the real systems allows to design and build unique services that can change systems analysis and management approaches, by shifted these process to a kind of augmented reality, where real systems are extended by their models and their models, like digital shadows, became their integral part. One of such solutions, called continuous modelling, described in [10], allows to perform monitoring of Internet of Things aware systems in parallel with their running models with constant comparison of expected state of the system with its actual state, what allows to quickly identify anomalies and deviations in system behavior.

Another joint processing of a model and real system data benefit can be achieved during creation of the model for an existing system that can be monitored in IoT-like way. When researcher is working on a new system or trying to define a new process, he can define both model structure (stocks, flows and their configuration) and equations he is going to put into his model as flows definitions. But not all works are starting from scratch. System Dynamics is frequently used to optimize and improve already existing systems which evolved over time without having their own models of any kind. In such cases, first action item which rises in the beginning of the work is analysis of the system with identification of its structure and understanding its behavior.

For example, frequent case is when we are having a running system and we need to understand how it behaves and what are the options we have to perform its optimization. When we start dealing with such situation, in majority of cases we do not have equations which will describe interactions between system components. In this case, to create a model we will need to reconstruct required dependencies via system monitoring and analysis.

II. Modelization definition

Today more and more system became a part of a paradigm called Internet of Things. In general, it means that components of such systems became connected to the Internet and can be managed and monitored using centralized services. New services, manufacturing, production and other systems became designed now to be IoT-enabled from scratch and in the same time many systems which are working for years already are enhanced to be IoT-aware with wide range sensors and different feedback processing devices allowing influence

ISSN 2305-7254

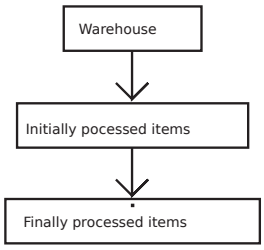

Fig. 1. Basic structure of a demo model with two flows and three stocks

systems remotely. An easy example is equipping production lines with monitoring sensors, and remote power management units allowing to start, stop and monitor production lines using a centralized service.

Since new technologies are coming to scene, questions about new ways of system optimization are bubbling up. Now with all new monitoring possibilities, it is reasonable to try leveraging them to achieve better productivity and reduce various kinds of waste.

Another aspect to take into account is growing computation power and computation accessibility. Today we can perform much more computations than even few years back and this leads to growing application of all kinds of computer modelling to support almost any human activity.

Joining together these two features, gathering of IoT data and application of models for system understanding, monitoring and improvements, can bring lost of benefits for most of industries. In terms of this article, we will focus on working with dynamic systems and under selected modelling approach we will assume System Dynamics toolkit.

To illustrate future discussion we will use a simple model that consists of three stocks and two flows. For example, such model may represents a supply chain pipeline where we have production warehouse, consumer one, that process items from this warehouse and consumer two that perform a final processing over results of first-line processing made by consumer one. Fig.1 schema depicts described model.

In terms of this articles, under system modelization, we will understand a process of automated definition of equations representing system dynamics model flows by analyzing stocks monitoring. The basic idea of this approach can be explained in the following way:

1) Model structure definition. During this step should be defined key stocks (nodes of the model) and flows. Here we are interested only in the structure of the model without its functional aspects, and it will be enough to define source and destination stocks for each flow without specifying any equations explaining flow behavior.
2) Data acquisition. At this stage, we will perform monitoring of the target and will retrieve the state of each system node. As a result of this stage, we will get a list of aligned time series data sets describing a state of each model flow over time. It is important to have this time series aligned in time with the precision that makes sense for the concrete use case. For example, if we are talking about supply chain model, several minutes drift between time series will not be critical, but if we deal with the model of a short running process like cooling down a teacup - several minutes will be the time of the whole experiment.
3) Monitoring data analysis. After all time series are received and aligned in time, they should pass through a chain of statistical processing methods like an approximation. The goal of this step is to find an equation that will describe changes of stock (node) state over time or of flow intensive.
4) Restoration of non-measured data. In practice, during the experiment, we can't get all data of a system, for all stacks and flows because technical or economic reasons can't measure some of them. The goal of this step is to find an equation the will describe changes of stock state and flow intensive which we can not measure directly.
5) Flows equations reconstruction. This step is a key element of the proposed schema. Taking into account the behavior of each model stock as functions from the time, we will try to reconstruct flows, which are linking stocks which behavior we just investigated and described analytically with equations we extracted from system monitoring results.

It is important to understand, that in most cases, it is not possible to achieve the goal of complete and fully automatic reconstruction of system dynamics data from real system monitoring, especially for complicated systems with big numbers of flows going back and forth between stocks. Goal of the proposed approach is to design and develop an instrument that will help to reconstruct a model of the system specialist needs to work with, buy providing him advanced and enhanced view of the monitored system with the number of recommendations that will help him to solve this problem.

Whole system modelization algorithm can be split on two phases: defining model structure, setting up monitoring, retrieving monitoring data and processing them to get analytic function definitions instead of pure time series. Second phase will be an attempt to reconstruct a model from obtained and processed data.

First part of this algorithm is sown on fig. 2

III. Model structure definition

This is a first step, in building a model of existing system. When passing this step, it is important to understand that model complexity, will affect complexity of the rest of the process and higher will it be - more complicated computations will be required and more assumptions will be needed to be made on further process stages.

While working with model structure it is better to follow incremental processes like ones described for software development in Agile methodologies. Process of model reconstruction can be treated as an incremental detalization and improvement

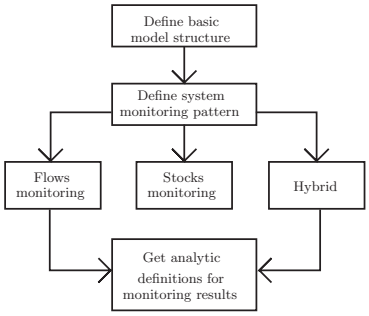

Fig. 2: First phase of modelization algorithm

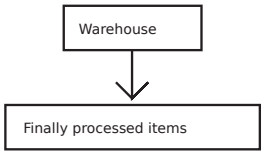

Fig. 3. Simplified structure of a demo model with two stocks and only one flow

of the model we are working on with step-by-step diving into the system under analysis.

Getting back to our example with a production pipeline with two sections, in a very first version of the model we may define it at a glance like a two step model, like shown on fig. 3.

In this case, whole processing will be treated as a one single operation without dividing it into processing stages. Such simple model will allow to understand overall system behavior. When on such level all required model components will be reconstructed, this model can made more complicated and intermediate processing stage can be revealed, like it was shown on initial fig. 1. Whereat can be revealed sub-steps of each processing step and so on, until model will not reach required detalization level to address issues it is designed for. Each define d stock will require performing monitoring and analysis of monitoring results. For each defined flow we will need to pass all steps to reconstruct flow equations from the monitoring data retrieved from related to it monitoring flow.

IV. Data acquisition

After defining model structure, next important step is proper acquisition of model data. Since we are dealing with real systems, it is not always possible to properly measure states of all system components we are going to include in the model. There are two key measuring cases, how system component can be monitored: state monitoring and flows monitoring.

In case of system dynamics model, model nodes, represented by stocks are not functional and they can be treated just like a storage accumulating some entities. In the same time flows are processes which are changing values of the stocks. In other words, everything what performs actions on stocks values is a flow. For example, a truck which is moving a cargo from one storage to another is a flow connecting two warehouses (stocks). Another example is furniture factory, it

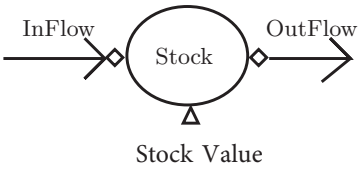

Fig. 4. Stock state monitoring options

consumes wood from one place, produces furniture and puts it in another. In this case, factory is a flow and on different sides of this flow i t c onnects t wo s tocks w ith c ompletely different entities.

When we are performing system monitoring, in most cases we can monitor only state of the stocks, but not flows directly. On fig. 4 shown two options for stock monitoring.

If architecture of the system allows to measure stock input and output flow, i t m eans, t hat w e f aced a r elatively easy case, because during system monitoring we knows, direction (increasing or decreasing) and pace at which stock value is changed. This is already vary close to the solution we are trying to achieve - definition o f t he fl ows in th e running system.

In most cases, it is not possible to properly measure input and output flows and the only metric that can be monitored is stock actual value. In other words, lets say, that we have scales with sugar and there are two flows: one adding sugar to the scales and other is taking it from there. These two processes are performing simultaneously and the only thing we know at any given moment is amount of sugar on scales. In this case we will not be able to directly define properties of each flow. To this topic we will get back later in chapter **??**

V. Monitoring data analysis

Next important step is data analysis. In the current context, under data analysis, we will assume processing of monitoring data we got from the system under analysis.

As a result of monitoring, we will receive a time series describing required system components which are represented by stocks in the model. From processing perspective it is not always easy to work with time series directly, also, taking into account that our goal is to fill gaps in the model definition with filling in, it required to convert this data series into analytic representation. There are several different approaches allowing to find a function that will approximate experimental data.

Approximation of experimental data is a topic that is well described in many sources like [4] or [5]. The goal of approximation is to define an analytic function that will go as close as possible to the set of experimental data. When approximation is performed we need to do key things: first, define a function that will be used for approximation, for example linear, polynomial, exponential, trigonometric and so on, and second - define proper configuration and coefficients of this function.

Often, when a researcher or analytic receives experimental data, he can try guessing the best approximation function by looking at experimental data charts. In other words, based on

his own unique experience, concrete case and a set of available data, he is making a decision about the shape of the function, that will give him closest approximation results.

When designing cloud service solution, it is not possible to address approximation aspect this way, because system under development should be tolerant to different sequences of input data received for analysis and these data sets should be processed with some defined precision and quality level. To address this problem in an automated way, there is a couple solution.

The first approach that can be used is a multivariate parallel approximation. For that purpose, on a platform level, we will define a set of most popular functions that can be used for approximation purpose. For example, we may have linear, polynomial, exponential, logarithmic and trigonometric approximation bases in the library. When new data set received for analysis, the system will do a parallel approximation using each of bases available in the library. Quality of approximation can be measured, for example, by using standard deviation between experimental data and found approximation function. When all parallel approximations are completed, we can make a decision about best-found approximation using selected approximation quality metric. The key benefit of such solution will be the ability to easily extend library of approximation bases with new entities and with the ability to update existing ones with better computation methods, because, the polynomial approximation can be performed with the number of was, for example using a method of Least Squares or using Tailor series. The principal disadvantage of such an approach will be high consumption of computation resources. When designing a cloud solution, we have to assume, that there can be a significant number of researches, pushing data to the service for analysis and all of them are expecting responses in a reasonable time. The approximation is an expensive job from resources consumption perspective and performing it in parallel with a big library of approximation bases can give very good results, but will take much time. This method can be made cheaper, if we will allow specifying the number of patterns researcher expects in his data, this will lead to decreasing number of required computation, but will require knowledge about time series passed to for analysis.

The second approach that might help to address the issue with defining closest approximation function shape is an application of machine learning algorithms. Machine learning works well for different classification jobs. Definition of time series shapes is a typical classification problem where we need to identify the ranked set of most suitable shapes for the given time series.

Deep learning has been used successfully for time series prediction. In particular, recurrent neural networks (RNNs), especially those utilizing long short-term memory (LSTM) nodes, are useful for sequential tasks like this. Lipton, Kale, and others [6], have used LSTM RNNs for time series of medical data. They demonstrated that an RNN can ingest a sequence of clinical medical observations and then accurately classify many diseases based upon these time series of raw sensor readings and lab tests. Having a library of expected time series shapes and trained neural networks will allow reducing the number of parallel approximations we need to attempt to get the result, closest to the sensor data processed.

Let describe what we have after this step from the mathematical point of view:

- Functions (F, G, H...) of all measurement stocks from the time ($F : time \rightarrow value$).
- Function of stock value change for a specific time interval ($F' : time \rightarrow valuechange$). Also, in non-strict means, we can call it a derivative of a stock function.
- Functions of all measurement flow influence to source and destination stocks ($f : time \rightarrow valuechange$ and $f^{-1} : time \rightarrow valuechange$). It should also be noted that this function is not a flow equation, because the flow is a function of stock value, not from the model time.

VI. Flows equations reconstruction

This is a key step of the described approach. Goal of this step is to attempt to recreate equations, describing each flow of the model which structure was defined in the beginning of modelization process.

Let formalize a problem that should be solved on this step. We already have an influence function of most flow in the model, but as noted above, it's not a flow definition, because flow is a function from stocks values $f' : (stockvalue1, stockvalue2...) \rightarrow valuechange$. At this step we should replace all flow influence function to it's flow equivalent in regarding to the follow term:

$$f(t) \equiv f'(F(t), G(t)...)$$

There are number of options to use for flows functions reconstruction and approach selection will depends on what kind of monitoring was possible to perform on a system under analysis.

1) Input and output flows monitoring. Let's say that for a pair of stocks (or system components) we were able to perform monitoring of output flow for the source stock and input flow of destination stock. It means that for this particular flow we have its input and output values. Since flow, in terms of model, is a pipeline that transport entities from one stock to another at a certain speed, without changing quantity, it will be enough to know values at any end of the pipeline, because on the other side, flow value will be the same, but with opposite sign. Keeping in mind that flow is representing a rate at which stock value is changing and the thing that following the described approach for turning monitoring results into analytic representation gives us an equation describing stock changes over time, we can get a flow value by getting derivative of a stock function.
2) Stocks monitoring. This case is much more complicated comparing to processing results of monitoring for input and output flows. Key issue is that from monitoring result we got is a state of the stock in each moment in

time during monitoring, but this doesn't give us information about influence of input and output made by flows connected to the stock. The only thing which is known for us is a result of a function which is a superposition of input and output flows of the stock. In other words, at any moment t_1 value of the stock can be represented as $F'(t_1) = \sum_f^{InputFlows} f(t) - \sum_f^{OutputFlows} f^{-1}(t)$. Order of functions superposition is depending on model structure, because, according to System Dynamics models specification, flows execution order depends on their order definition in the model. Situation with stocks monitoring can became even worse, if a stock has multiple input and output flows, what is common in some cases, when model components are tightly couple to each other. Such situations are often seen in economical and environment-related models. But, there are also cases, when stocks monitoring works well. This is the case when we are dealing with leaf-nodes of the model and these nodes have only one input or output flow. In this case stock monitoring can be treated as a input or output flow monitoring and we will come to the first case.

3) Hybrid mode. Hybrid mode means that some of the stocks are monitored directly and others are monitoring on a level of their input and output flows. This approach inherits all advantages and disadvantages of both described approaches, but, it can bring benefits upon identification of system monitoring strategy. For example, we have a system, which consists of three jars of some liquid. These three containers are connected in chain via two filters cleaning liquid on the way from then first jar to the third one. Dealing with liquid flow measurement is non-trivial thing, comparing to measurement of its volume. That is why, for the first and third jars we can use simple volume (or weight measurement liquid density is known) monitoring for the first and third jars and perform flows monitoring only for the second one which is affected by superposition of input and output flows.

VII. Restoration of non-measured stocks and flows

In some cases, modelization process can be enhanced via model simplification process. Interms of this section, under simplification we assumes not optimization of the model itself, but incremental cut-off process on a way of model flows incremental definition, where we are dropping already reconstructed flows and continuing processing only the rest of the model, until we face a tightly couple model piece that can be processed any further.

Basically, flows are describing a rate at which this particular flow changes connected stock value. So if a stock has multiple input and output flows, its value at every moment can be calculated as a sum of all itput flows minus sum of all output flows, like this:

$$F'(t) = \sum_f^{InputFlows} f(t) - \sum_f^{OutputFlows} f^{-1}(t)$$

In a corner case, if a stock has only one input or output flow, we can easily reconstruct its flow by getting a derivative of a function that represent stock value change over time. In this case such stock and easy reconstruct-able flows can be solved and removed from the model, to leave smaller structure for further processing.

This recurrent and incremental process ends when there will be no single-flow stock left int he system. This procedure can be described with the following algoritmh:

1) Find all unambiguously defined flows by leaf stocks. If we know the stock function and only one flow will be connected to it, we also know that this flow has an identical influence on it, and it is possible to define that $F'(t) \equiv f'(t)$.
2) Find all unambiguously defined leaf stocks by flows and processed as in the previous step.
3) Simplification. Analysis all known data and remove influence of all known flows by a function subtraction. For example: we have three stocks F, G, H, and two flows f (from F to G) and g (from F to H). We have measured data of F and G stocks. By the first step of this algorithm we can define f flow influence function. We know, that $F'(t) = -f(t) - g(t)$ and we can find the g function by substitution of f from F'.
4) Repeat algorithm while we have any unambiguously defined stocks and flows.

At the algorithm stop's we have a simplified version of an initial system dynamic model, in which we can see only ambiguously flows and socks. Fig. 5 depicts described algorithm. If it is a not complicated model, we can go forward to the next modelization method step. But in the different case, the computation complexity may be too high to get modelization done in an acceptable time, and researcher should be to return to experiments and extends sensors environment.

VIII. Modelization example

To show how this method works on practice let's get back to the initial example shown on fig. 1.

Let's assume, that we have a simple production line which takes some item from an input storage, performs its initial processing, than, from initial processing results it is taken for final processing and after that it goes to the final production store.

When we started monitoring we know that at the beginning of production cycle we have 8 blank pieces that will be turned into products. In terms of this example blank pieces are converted into product 1:1.

When we started production monitoring we got stocks states distribution over time shown in table I. In this table S_1 a stock representing starting warehouse, S_2 is a box with initially processed pieces and S_3 is a stock with production pieces.

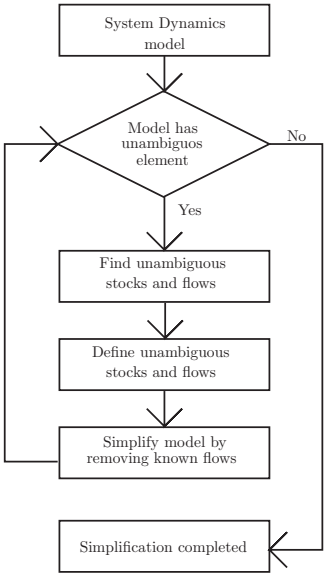

Fig. 5. Model incremental simplification

TABLE I. SAMPLE PRODUCTION PIPELINE MONITORING RESULTS

Time	S_1 value	S_2 value	S_3 value
0	8	0	0
1	6	2	0
2	4	3	1
3	2	4	2
4	0	5	3
5	0	4	4
6	0	5	5
7	0	2	6
8	0	1	7
9	0	0	8

From our model definition we see that stocks S_1 and S_3 are affected by only one stock. It means that we can calculate flow value for them directly based on monitoring results. Obviously for this simple case, both approximations will be linear functions $y_{S1} = -2x$ and $y_{S3} = x$ after first item passes though the model. According to the definition of flow in system dynamics, it is the rate at which the stock is changing at any given instant, they either flow into a stock (causing it to increase) or flow out of a stock (causing it to decrease). From them math we know that change rate for the function is function derivative. In our case, flows will turn into -2 for S_1 out flow and +1 for S_3 in flow. Other sides of the flows connected to S_2 will have the same values, but with opposite sign, because nothing is disappearing from the flow and nothing new appears.

To see some corner cases, let's use a bit more advances example. Assuming that we have one more processing step between stocks S_2 and S_3 and monitoring results appeared like they are shown in the table II

Here we can see, that we have two flows which are affected stocks S_2 and S_new. Monitoring data shows, that everything what get's into the new stock - immediately sent to the S_3. Input flow which comes from S_1 to S_2 is known, it is $+2$.

TABLE II. EXTENDED SAMPLE PRODUCTION PIPELINE MONITORING RESULTS

Time	S_1 value	S_2 value	S_{new} value	S_3 value
0	8	0	0	0
1	6	2	0	0
2	4	3	0	1
3	2	4	0	2
4	0	5	0	3
5	0	4	0	4
6	0	5	0	5
7	0	2	0	6
8	0	1	0	7
9	0	0	0	8

Based on this and values of $S2$ that was measured, we can figure out, that output flow from S_2 is -1. But the same technique will not work for the S_{new}. We expect, that input flow is +1, but, we can estimate an output one. at minimum, it is equal to the input flow, but we are not able to estimate its maximum. This example gives us two important outcomes:

1) Modelization process do not describe system properties and actual characteristics. It allows to formalize actual current system behaviour. Complete modelization process may require execution of system under different workload to see performance and boundaries of all flows.
2) During the modelization it is easy to identify flows where overflow (situations when input flows of a stock exceeds output flows) or underflow (situations when output flows of a stock exceeds input flows, like on the second example). In many cases, detection and addressing of such situations in one of the most important cases during system tuning and optimization.

IX. METHOD EVALUATION APPROACH

Described method was tested on synthetic data sets and shows its efficiency in achieving stated goals. As a baseline for the experiments was taken a library of test system dynamics test models which is used for evaluation and testing of various computation engines developed and maintained by tools community of System Dynamics society [7]. As a models execution engine was selected engine called PySD [8], because it is developed on a Python language and it appears faster and easier to make code modifications for method evaluation purposes than in other available execution engines. Evaluation and testing of proposed approach was performed on facilities on sdCloud project [9].

Experiment was performed on 50 different models and was designed in a following way.

1) Basic model execution with advanced logging. on this step we executed 50 different test models using modified execution engine. Goal of the modification was to log values of all flows before entering or right after leaving model stocks. This gives us extended model execution results which contains not only time series describing stocks history, but also input and output flows value for each stock.

2) After execution is completed, from the model definition excluded all equations representing flows. This turns a model into pure structure model that contains only frame of the model.
3) All time series describing stocks values from modelling results are converted into closes analytic representation. According to the approach definition both approaches were tested: with simple parallel approximation using different functional patterns and selection of best option by estimating approximation quality and with application of trained neural network for time series shape detection for future approximation using top three detected shapes.
4) Since we had all input and output flows (from extended model execution logs), for each model we were able to reconstruct its flows via definition of approximation function based on logged input and output flow values

X. Method evaluation results

As a result of our experiment was found that for majority of models, 89% of flows across all test models, were recovered correctly. Correctness of the reconstruction of each flow was identify by comparing flows equations in original and newly created model. Flow reconstruction was count as successful if correct flow shape was used without taking into account contestant term and allowing slight deviation of coefficients for lower power terms.

Parallel approximation shows a bit better results in selection of approximation function comparing to application of neural network-based classifier, 89% against 75%. But second option shows much better results in performance and resources consumption. Most probably, this gap can be reduced in case of better training of neural network.

Draft implementation of proposed feature was implemented within sdCloud platform application and integrated with the service for continuous modelling, to leverage its mechanisms for accepting input data streams for further processing within the platform.

This work was supported by grant 18-57-45004 IND_a from RFBR.

XI. Conclusion

In this article was described an approach aimed to help building System Dynamics models out of existing system by integrating system monitoring, data analysis and model execution practices. Proposed approach significantly depends on abilities for monitoring of target system and has number of limitation, but for many system dynamics application areas it can significantly improve modelization process and make it easier and faster than manual system investigation.

Proposed method is trying to leverage both classical statistics methodology and modern machine learning mechanisms allowing to solve some issues with the higher run-time performance and less resources consumption, but, in the same time requiring special efforts on a preparation step, like training of neural networks.

References

[1] *B*usiness Dictionary, Web: http://www.businessdictionary.com /definition/system-dynamics-SD.html
[2] *K*ermack WO, McKendrick AG (August 1, 1927). "A Contribution to the Mathematical Theory of Epidemics". Proceedings of the Royal Society A. 115 (772): 700–721. doi:10.1098/rspa.1927.0118
[3] *H*ethcote H (2000). "The Mathematics of Infectious Diseases". SIAM Review. 42 (4): 599–653.
[4] *S*mirnov, V.Y., Kuznetsova, A.V., Approximation of experimental data by solving linear difference equations with constant coefficients (in particular, by exponentials and exponential cosines), Pattern Recognit. Image Anal. (2017) 27: 175. https://doi.org/10.1134/S1054661817020109
[5] *B*elikovich, V.V., Approximation of experimental data by exponential function, Radiophys Quantum Electron (1993) 36: 831. https://doi.org/10.1007/BF01039696
[6] *L*ipton Z., Kale D., Elkan C., Wetzel R., Learning to Diagnose with LSTM Recurrent Neural Networks, arXiv:1511.03677
[7] *R*epository with System Dynamics test models: https://github.com/SDXorg/test-models
[8] *P*ython-based execution engine for System Dynamics models PySD: https://github.com/JamesPHoughton/pysd
[9] *s*dCloud - cloud-based platform for system dynamics modelling: https://sdCloud.io
[10] *P*enskoi A., Gaiosh A., Platunov A., Kluchev A. Specialised computational platform for system dynamics: ISBN 978-619-7408-39-3, ISSN 1314-2704, DOI: 10.5593/sgem2018/2.1, 2 July - 8 July, 2018, Vol. 18, Issue 2.1, 709-716 pp.